Hotspot Avoidance for P2P Streaming Distribution Application: A Game Theoretic Approach

Zhen Yang and Huadong Ma, *Member*, *IEEE*

Abstract—Today's peer-to-peer (P2P) streaming application periodically suffers from routing hotspots, which are also known as flash crowds. A routing hotspot is typically created by an unanticipated new event that triggers an unanticipated surge of users to request streaming service from some particular peers, temporarily overwhelming the peer's delivery capabilities. In this paper, we propose novel methods that avoid routing hotspots proactively, that is, prior to a congestion event. More specifically, we define an incentive-compatible pricing vector explicitly and show that the hotspot can be avoided if all nodes in the network follow the incentive-compatible pricing policy. In order to apply this mechanism to the P2P streaming distribution applications, we propose an adaptive algorithm for distributed computation of the incentive-compatible pricing vector. The simulation results show that the incentive-compatible pricing mechanism can avoid the routing hotspot effectively.

Index Terms—Hotspot avoiding, game theory, peer-to-peer, streaming service, incentive-compatible.

1 Introduction

Routing hotspots, also known as flash crowds, are among the most important problems that plague today's peer-to-peer (P2P) network [6], [11]. A routing hotspot is typically created by an unanticipated new event that triggers an unanticipated surge of users to request service from some particular peers, temporarily overwhelming the peer's delivery capabilities. It is reported in [5] that when such an unanticipated new event is created, nearly all responses come from the top 1 percent of information sharing nodes. Therefore, these responsive nodes are prone to congestion, which leads to the routing hotspot problem.

The high bandwidth demand of the streaming distribution applications [1], [2], [3], [4], [5], [6], [7], [8] in the P2P networks poses a significant technical challenge in resolving the routing hotspot problem. As nodes in P2P networks reside at the edge of the Internet, they usually have limited availability of upload and download capacities. Thus, it is difficult to overprovide the streaming distribution service for the peak demand [34]. Even if the bandwidth is sufficient at some peer, the users cannot justify the cost of overproviding the service for the peak demand and need cheaper alternatives to survive through transient bursts of massive request traffic.

Most studies on resolving the hotspot problem adopt a reactive approach [3], [4], [34], [35], [36], [37], [38], [39], in which the hotspot resolving process starts after a hotspot is detected. It usually takes quite some time to retrieve duplicates of recently generated objects. In this case, the difficult task is to provide good QoS provision for delay-sensitive media streaming applications.

At present, some works consider the proactive mechanisms [2], [31] about the single-source P2P media distribution and can provide good QoS provision. However, these mechanisms are no longer efficient for multisource streaming distribution. Some P2P streaming applications involve the multisource streaming distribution, such as video conferencing, multiplayer real-time game, and multisource VoD service. Thus, it is very significant to avoid the routing hotspot in the case of the multisource streaming distribution. In the multisource streaming distribution, the nodes in P2P network may be shared by multiple media sessions and thus are required to relay multiple media flows. In such a scenario, media sessions are greedy and selfish and do not follow socially accepted congestion control mechanisms. Greedy media sessions are willing to violate these mechanisms and obtain better performance of media distribution.

Considering these difficulties, we argue that, in P2P streaming systems, the hotspot resolving techniques should aim at the following objectives:

1. They should be able to anticipate the appearance of hotspots and to proactively divert the traffic off the congested node. In other words, the hotspot resolving process is performed when detecting a traffic surge and anticipating a flash crowd.
2. They should be simple and efficient to avoid the occurrence of hotspots at low cost. This is to say, hotspot resolving technologies can ensure high delivery ratios during periods of high traffic.
3. They should be stateless and hierarchy-free, facilitating implementation.

• *The authors are with the School of Computer, Beijing University of Posts and Telecommunications, No. 10, Xitucheng Road, Haidian District, Beijing 100876, P.R. China. E-mail: {yangzhen, mhd}@bupt.edu.cn.*

Manuscript received 21 Aug. 2007; revised 18 Feb. 2008; accepted 15 Apr. 2008; published online 23 Apr. 2008.
Recommended for acceptance by S. Das.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2007-08-0288.
Digital Object Identifier no. 10.1109/TPDS.2008.67.

4. They should be applied to both single-source and multisource media streaming distribution. Single-source scenario is a special case of multisource streaming distribution. When several media sessions (multisource media streaming distribution) share the same node, hotspot occurs often. Therefore, it requires us to find a fair/efficient way to share the available resources among sessions.

The goal of this paper is to design and evaluate hotspot avoidance techniques for quickly offloading the congested peers once the formation of a hotspot is detected. In a multisource media streaming distribution scenario, the problem lies essentially on incentives: How can we force selfish sessions to share the node resource equally and thus avoid hotspot as efficiently as possible?

In this paper, we consider the proactive hotspot avoiding mechanism based on game theory [31], [32]. In the P2P streaming distribution, a media session can select one or more relay peers for streaming the audio/video objects. We assume that each of the peers offers a dynamic and adaptive price. The price for using each peer has a congestion-sensitive pricing factor and is proportional to the congestion level at the peer. (In fact, such a price is already used by the KaZaA [2] system for quantifying a user's reputation and prevention of faked reputations. In that system, it is called the participation price.) As a result, when a peer is with high traffic, it will be granted a higher unit price than others. This increases resource utilization of each node and avoids the creation of hotspot. This paper makes the following contributions:

- First, we use the game theoretical method to study the hotspot avoiding technology in the P2P streaming distribution application. We model the P2P streaming distribution as a noncooperative game. We call this game as Streaming Distribution Game (SDG). Based on the SDG game, we also propose an Incentive-Compatible Pricing Mechanism (ICPM) as a means to avoid the routing hotspot.
- Second, we demonstrate how ICPM can be used to pilot the media distribution in P2P networking to a Nash equilibrium. In the context of our problem, a Nash equilibrium is a scenario where no selfish media session has any incentive to unilaterally deviate from the current hotspot avoiding strategies established by ICPM. It implies that following ICPM is the best strategy to avoid hotspot for each media session.
- Third, in order to apply ICPM to the real P2P streaming system, we propose an Adaptive Pricing Vector (APV) algorithm. It redirects the traffic from the congested node to its neighbors only based on the local information of the congestion node.

The rest of this paper is organized as follows: In Section 2, we review the related works. In Section 3, we formulate the hotspot avoiding problem. In Section 4, we present the properties of the Nash equilibrium imposed by ICPM. Section 5 presents the ICPM and investigates the structure of the incentive-compatible pricing vector. In Section 6, we introduce the adaptive algorithm and investigate its convergence property. In Section 7, we present the simulation result of the proposed mechanism. Finally, concluding remarks are given in Section 8.

2 Related Work

2.1 Routing Hotspot

Hotspot has been an active field of research over the past few years. Some hotspot resolving technologies have been explicitly designed to support massive demand and degrade gracefully under high load.

A large number of hotspot solutions are based the reactive approach [3], [4], [34], [35], [36], [37], [38], [39]. In such a hotspot solution, the hotspot resolving process starts after a hotspot is detected. For instance, the Flash Web server [35] combines an event-driven server for access to cached workloads with multithreaded servers for disk-bound workloads, to reduce resource requirements while increasing the capacity of the service. SEDA [36] relies on event-driven stages connected by explicit queues that prevent resources to become overcommitted when the demand exceeds service capacity. The JAWS Web server framework [37] supports multiple concurrency, I/O, or caching strategies to better customize the server to various deployment scenarios. The throttling mechanism [38] limits the number of simultaneous clients, or the bandwidth assigned per client; although such mechanisms prevent the server from failing, they provide degraded service to the clients under high load conditions. The NEWS system [39] protects nodes from flash crowds by regulating request traffic on the access router, based on observed response performance. Our hotspot solution is different from these technologies. First, we do not focus on file transfer but rather to aim at the P2P streaming distribution. Second, we do not adopt the proactive approach but rather to adopt the reactive approach.

Hotspot avoidance can also be achieved, albeit at a high cost, through combinations of software and hardware solutions such as traffic load balancers and clusters of servers [40], [41], [42]. For instance, the Google service [40] load-balances queries to one of multiple data centers, each hosting large clusters of machines. Commercial content delivery networks provide scalable Web service by replicating content over a large network of distributed servers and redirecting clients to local copies using proprietary algorithms added to the domain name system. However, the most traditional technique to avoid hotspots involves overprovisioning the source servers for peak demand, but given the unpredictable nature of flash crowds such preventive approaches require a good understanding of traffic patterns. In general, these solutions have a prohibitive cost and are mostly appropriate when the server is consistently operating under high traffic conditions and hosts critical information.

Several recent studies have tried to characterize the flash crowd phenomenon [43] and a few proposals have been put forth to leverage P2P architectures for tolerating such events as well as improving the scalability of content distribution [44]. SQUIRREL [43] is a client-side P2P Web cache that stores recently accessed documents on the client peers and looks for a reasonably fresh copy in the local caches before fetching a Web resource from the original

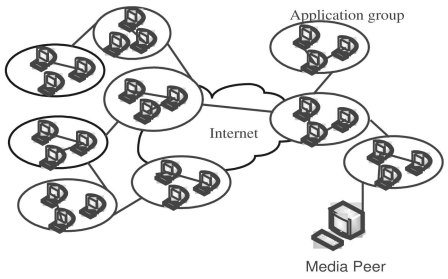


Fig. 1. The topology-aware overlay.

server. This Web cache does not operate transparently as it requires specific software to be installed on the client computers. PROOFS [44] is a P2P network that uses randomized overlay construction and scoped searches to efficiently locate and deliver content under heavy demand. When some content cannot be obtained from a server, peer clients try to obtain it from other peers instead. This approach does not heal the hotspots, and it requires specific software to be deployed on the clients. However, SQUIRREL and PROOF target the Web service. Our work, on the other hand, aims at P2P streaming applications.

Some works consider the proactive mechanisms about the single-source P2P media distribution. For example, Xiang et al. [2] introduce a novel framework for multimedia distribution service based on topology-aware P2P networks, called Topology-aware P2P Media Distribution Network (TPMDN) framework. In the TPMDN, peers self-organize into application groups (AGs; see Fig. 1). End hosts within the same group are with similar networking conditions and can easily collaborate with each other to achieve the QoS awareness. In order to improve the quality of media delivery and provide high service availability, the authors further propose two distributed heuristic replication strategies, intergroup replication, and intragroup replication. TPMDN framework can avoid hotspots to some extent. However, TPMDN framework considers the case of single-source streaming distribution. On the other hand, we consider the case of both single-source and multisource streaming distribution.

2.2 Incentives in P2P System

In order to understand the background of our research, we review some related works using the incentive mechanism.

Game theory [32] is a mature theory and the formal study of conflict and cooperation. Game theoretic concepts apply whenever the actions of several players are interdependent. These players may be individuals, groups, firms, or any combination of them. The concepts of game theory provide a language to formulate, structure, analyze, and understand strategic scenarios.

The current challenges of game theory applied to computer networks are summarized by Papadimitrou [33]. Several papers (for example [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]) have applied the incentive and price approach to computer networks over the last 15 years. A thorough literature survey is beyond the scope of this paper. We now review some of the most directly related work.

As an allocation mechanism, pricing mechanism has been studied in the context of queuing systems, e.g., [15] and [16]. Some proposed schemes are pricing for resource reservation (through access control) [25], pricing for priority [17], pricing for a given transfer rate [18], or auctioning [19]. Pricing mechanisms for QoS provisioning can be classified into static, dynamic, or auction-based schemes. In the case of dynamic pricing, the network adapts prices as the traffic load changes. In auction-based pricing schemes, users attach a bid to each packet indicating the willingness to pay for the delivery of the packet, and the network serves packets in descending order of their bids.

Generally, static pricing schemes are simpler to implement compared with dynamic or auction-based schemes. However, this ease of implementation comes at a cost as prices have to be chosen a priori and it is not possible to react to changes in the users demand or traffic load in the network.

Dynamic pricing is the most powerful and flexible mechanism: it can be used to achieve a socially optimal bandwidth allocation, maximize profit, and obtain an incentive-compatible class allocation [15], [16], [17]. Thus, in this paper, the ICPM is dynamic pricing.

Auction-based pricing mechanisms are extensively studied, because of their incentive properties [19], [20], [21], [22], [23]. For example, MacKie-Mason and Varian [19] suggested the use of a smart market, a per-packet auction to solve congestion problems through pricing. In [23], the authors designed the so-called multibid auction pricing mechanism, which is a one-shot auction-based scheme to share the capacity of a single communication link among several users. They proposed that each user submit several bids when establishing a connection. And, they also proposed that the corresponding multibid profile be used to compute efficient allocations and prices. The multibid scheme has been shown to verify several desirable properties in terms of incentives and efficiency. An advantage of the multibid scheme is that it does not require any synchronization among users: A user may bid at any time (without knowing the bids of other users), potentially modifying the allocation and price of other users until the end of its session. However, there is no guarantee of keeping an initial allocation during a whole session. Furthermore, the efficiency of those schemes is not established analytically. Moreover, multibid scheme auctions need a phase for the submitted bids, implying time burden and signaling overhead. Thus, these multibid auctions only consider the case of file transfer application. In our work, we propose an adaptive algorithm to reallocate the resource iteratively for the delay-sensitive media streaming distribution.

3 The Streaming Distribution Game: Preliminaries

In this paper, we consider a P2P streaming application with single or multiple media sessions. Each media session has one streaming source and multiple participating receivers. We model the P2P streaming distribution as a noncooperative game. The game is called SDG. In a game, there are rules and players. In the SDG, the rules are set by an

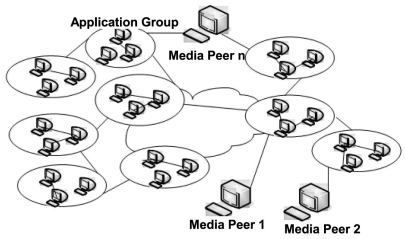


Fig. 2. Network model.

ICPM and the players are the media sessions. These sessions are selfish, that is, they are only concerned about their own good. Each player has a strategy, which is to control flow configuration that the player tries to push through the network.

3.1 Network Model

A streaming system is required to encompass the key functions of object lookup, peer-based aggregated streaming, and dynamic adaptations to network and peer conditions. In this paper, we reform TPMDN (see Section 2.1) as a sample system that satisfies the requirements of a streaming system.

The TPMDN considers the case of single session P2P streaming. In this paper, we consider the multisource P2P streaming. Thus, we extend the TPMDN model and proposed the Multisource Distribution Network Model (MDNM) for the P2P streaming application. Similar with TPMDN model, application based on the MDNM is QoS-aware and cost-effective. However, the MDNM can be applied to both single-source and multisource streaming distribution.

As shown in Fig. 2, there coexist multiple media peers as streaming sources in the MDNM. Each streaming source initializes one media session. Similar with TPMDN, a media session in MDNM is established as follows: Peers requesting a media file issue the lookup request to the underlying P2P substrate, which will return a set of candidate peers who have the content. Usually, a media session requires more suppliers than a file downloading session, which implies more cooperation among users is required in a streaming environment. The candidate set of MDNM typically contains 5 to 10 peers. A topology is constructed to connect the candidate peers with the receivers.

In such a topology, nearby hosts or peers are clustered into AGs. To maximize the efficiency of a media streaming session, every collaborative peer in an AG is willing to transfer media streams to the requesting peers. Since peers within one AG are very close to each other, the QoS requirements for content delivery, such as latency, can be easily satisfied. Moreover, to achieve the load balance and avoid hotspots, each session ships its stream by splitting its demand over the nodes of the same AG. In fact, the single-source media distribution is the special case of the multisource media distribution. Thus, the multimedia distribution service based on the MDNM is very similar to the case of TPMDN. Due to space limitation, we only describe the network model and do not cover the detail about multimedia distribution service (see [2]).

Due to the inherent real-time requirements of media streaming applications, the hotspot solution should be effective in a timely manner. We need some incentive mechanism to urge the media sessions to save the resources, and if they do not save, they should suffer from a poor quality of streaming delivery.

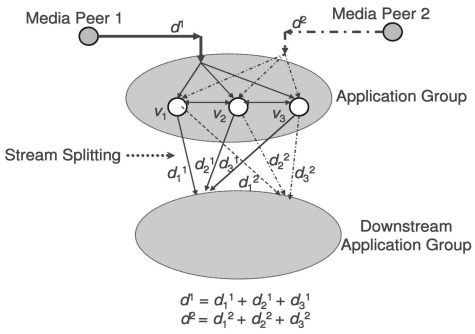


Fig. 3. Two sessions sharing an AG.

3.2 Problem Formulation

Based on the MDNM, we consider the scenario that a set of media sessions, i.e., $S = \{s^i | i = 1, 2, \ldots, R\}$, share one AG $V = \{v_j | j = 1, 2, \ldots, M\}$. Each session s^i has a flow throughput demand $r^i > 0$. Without loss of generality, we assume that $r^1 \geq r^2 \geq \cdots \geq r^R$. The session s^i ships its stream by splitting its demand r^i over the group V. A session is able to decide (at any time) how its demand is split among the nodes in the AG, i.e., the session s^i decides what fraction of r^i should be relayed through each node in the AG. Fig. 3 shows the scenario that two sessions sharing an AG split their streams over the group to avoid the hotspot. Let d^i_j denote the expected flow that the session s^i sends on the node v_j. Thus, the session s^i can fix any value for d^i_j, as long as d^i_j satisfies the following: 1) nonnegativity constraint: $d^i_j \geq 0$ and 2) demand constraint: $r^i = \sum_{j=1}^{M} d^i_j$.

Now, we turn our attention to a node $v_j \in V$. Let c_j denote the bandwidth capacity of the node v_j. Let d_j be the total flow on the node v_j, i.e., $d_j = \sum_{i=1}^{R} d^i_j$. And, d_j also denote the vector of all session flows in the node $v_j \in V$, i.e., $d_j = (d^1_j, d^2_j, \ldots, d^R_j)$. The flow configuration d^i of the session s^i is the vector $d^i = (d^i_1, d^i_2, \ldots, d^i_M)$. The system flow configuration $d = (d^1, d^2, \ldots, d^R)$ is the vector of all session flow configurations and takes values in the strategy space $D = \bigotimes^{s_i \in S} D^i$, where D^i denotes the set of all feasible d^i, and D denotes the set of all feasible d. We say that a session flow configuration d^i is feasible if its components obey the nonnegativity and demand constraints. Similarly, a system configuration d is feasible if it is composed of feasible session flow configurations.

In this paper, we measure the performance of a session s^i by a cost function $J^i(d)$. For a session, it acts in a selfish manner and aims to find a strategy d^i that has good maximization. Since the cost functions depend on the flow configuration of all sessions, it turns out that the optimal decision of each session depends on the decisions made by other sessions. We denote the decisions made by other sessions as $d^{-i} = (d^1, \ldots, d^{i-1}, d^{i+1}, \ldots, d^R)$. Due to the session being selfish, the SDG is a noncooperative game. Thus, we are interested in the Nash solution of the game. In other words, we seek a system flow configuration such

that no session finds it beneficial to change its flow on any node. Formally, a feasible system flow configuration $\widehat{d} = (\widehat{d^1}, \widehat{d^2}, \ldots, \widehat{d^R})$ is a Nash Equilibrium Point (NEP) if, for all $s^i \in S$, the following condition holds:

$$\widehat{d^i} \in \arg\min_{d^i \in D^i} J^i(d^i, \widehat{d}^{-i}), \quad \text{where } i = 1, 2, \ldots, R. \tag{1}$$

The NEP concept is of significant importance from a dynamic standpoint. In a practical scenario, a session changes its flow repeatedly in response to the varying load conditions of the nodes in the same AG.

In order to incentive the session to rationally split its flow and thus avoid the hotspot, we assume that the price of a node is proportional to the level of congestion at the node. The level of service provided by the node v_j depends on the total flow offered to the node v_j, i.e., $d_j = \sum_{i=1}^{R} d_j^i$, and is quantified by means of a congestion function $G_j(d_j)$. The higher $G_j(d_j)$ is, the lower the level of service provided by the node is. In this paper, $G_j(d_j)$ is interpreted as the average delay on the node v_j. We now present the congestion functions as follows:

$$G_j(d_j) = \begin{cases} 1/(c_j - d_j), & \text{if } d_j < c_j, \\ \infty, & \text{if } d_j \geq c_j. \end{cases} \tag{2}$$

Correspondingly, the price for using the node v_j is a function of the total flow d_j carried by this node. Therefore, we define the pricing function as follows:

$$P_j(d_j) = \begin{cases} \psi/(c_j - d_j), & \text{if } d_j < c_j, \\ \infty, & \text{if } d_j \geq c_j, \end{cases} \tag{3}$$

where the constant ψ_j can be interpreted as the congestion-sensitive pricing factor (weight) of the node v_j. ψ_j determines the relative significance of the congestion level at the node v_j. Similarly, the vector $\psi = (\psi_1, \psi_2, \ldots, \psi_M)$ is the congestion-sensitive pricing vector.

According to the pricing function $P_j(d_j)$, the cost incurred by the session s^i under the configuration d should be

$$J^i(d) = \sum_{j=1}^{M} d_j^i \times P_j(d_j) = \sum_{j=1}^{M} d_j^i \times \frac{\psi_j}{c_j - d_j} = \sum_{j=1}^{M} J_j^i(d_j). \tag{4}$$

According to (4), the cost function J^i has the following properties:

- P1: J^i is the sum of cost for the session s^i in every node of the same group, i.e., $J^i(d) = \sum_{v_j \in V} J_j^i(d_j)$.
- P2: $J_j^i : [0, \infty) \to [0, \infty)$ is a continuous function. J_j^i is a strictly increasing function of two arguments, namely the flow of session s^i and the total flow on the node v_j, that is, $J_j^i(d_j) = J_j^i(d_j^i, d_j)$.
- P3: J_j^i is convex in d_j^i.
- P4: Wherever finite, J_j^i is continuously differentiable in d_j^i, we denote $K_j^i = \partial J_j^i / \partial d_j^i$.
- P5: For every d, if not all costs are finite then at least one session with infinite cost ($J^i(d) = \infty$) can change its own flow configuration to make its cost finite.

Our goal is to enforce an NEP $\widehat{d} = (\widehat{d^1}, \widehat{d^2}, \ldots, \widehat{d^R})$ that is considered efficient from the viewpoint of hotspot avoidance. $\widehat{d}$ is the point to minimize the overall congestion in the AG. The Nash equilibria of the SDG depend on the pricing mechanism (i.e., the pricing function $P_j(\cdot)$). The pricing mechanism to induce a unique Nash equilibrium is defined as follows:

Definition 1. *A pricing mechanism with the pricing vector ψ is called incentive compatible if it induces a unique Nash equilibrium $\widehat{d}$.*

Depending on the way the network is financed and the market structure, we might determine prices according to various objectives that combine, in general, the aim to operate the network efficiently and, at the same time, generate revenue. In determining prices, we should also take into account the structure of the user community, as well as the competition by others. We do not attempt to address all these issues here. Rather, we aim at a rigorous investigation of pricing strategies that lead to efficient utilization of network resources.

An effective pricing mechanism should discourage the sessions from utilizing congested node resources and thus avoid the creation of the hotspots. A simple pricing policy to achieve this goal is to set the right pricing factor for each node in the AG. Hence, we concentrate on the computing of pricing factor. In other words, in order to drive the sessions to a Nash equilibrium and avoid the routing hotspot, we need to seek the right pricing vector ψ. Similar to Definition 1, any such ψ will be called an incentive-compatible pricing vector. In the following sections, we will show that for the NEP $\widehat{d}$, an incentive-compatible pricing vector $\widehat{\psi}$ exists, that is, an ICPM exists.

4 Properties of the Nash Equilibrium

In this section, we investigate the properties of the Nash equilibria imposed by ICPM on selfish sessions. The existence question is answered in Section 4.1. The uniqueness question is answered in Section 4.2. The monotony is given in Section 4.3.

4.1 Existence of the Nash Equilibrium

Because of the properties of the cost function described in the previous section, the SDG is equivalent to a convex game in the sense of [19]. Thus, the existence of an NEP is guaranteed. We have the following theorem:

Theorem 1. *In the SDG, the cost function of each session is defined by (4), the existence of an NEP is guaranteed.*

A proof of Theorem 1 is provided in Appendix A.

The minimization in (1) is an optimal solution. The optimal solution is equivalent to the following Karush-Kuhn-Tucker (KKT) conditions. For every $s^i \in S$, there must exist a Lagrange multiplier λ^i such that, for every node $v_i \in V$,

$$K_j^i(d_j) = \lambda^i \quad \text{if } d_j^i > 0, \tag{5}$$

$$K_j^i(d_j) \geq \lambda^i \quad \text{if } d_j^i = 0. \tag{6}$$

In other words, the KKT conditions are the necessary and sufficient conditions for a feasible system flow configuration d to be an NEP.

4.2 Uniqueness of the Nash Equilibrium

Given the existence of an NEP, we now investigate its uniqueness. The following result implies the uniqueness of the NEP imposed by pricing mechanism on selfish sessions:

Theorem 2. *In the SDG, the cost function of each session is defined by (4), the NEP $\widehat{d}$ is unique.*

A proof of Theorem 2 is provided in Appendix B.

4.3 Monotony of the Nash Equilibrium

Now, we investigate the monotony of the Nash equilibrium. Throughout this section, we consider renaming the nodes in the group in decreasing order of their normalized capacities, that is, we assume that

$$c_1/\psi_1 \geq c_2/\psi_2 \geq \cdots \geq c_M/\psi_M. \tag{7}$$

The following theorem implies a number of monotony properties of the Nash equilibrium:

Theorem 3. *Let $\widehat{d}$ be the unique Nash equilibrium of the SDG with the capacity configuration c and the incentive-compatible pricing vector $\widehat{\psi}$ to satisfy (4), we have the following conclusions:*

1. *For any session s^i, the normalized flows are decreasing with the ordering number of nodes, i.e., $\widehat{d_1^i}/\widehat{\psi_1} \geq \widehat{d_2^i}/\widehat{\psi_2} \geq \cdots \geq \widehat{d_M^i}/\widehat{\psi_M}$.*
2. *For any node v_j, the flows decrease with the ordering number of sessions, i.e., $\widehat{d_j^1} \geq \widehat{d_j^2} \geq \cdots \geq \widehat{d_j^R}$.*
3. *The equilibrium prices are increasing with the ordering number of nodes, i.e., $\widehat{\psi_1}G_1 \leq \widehat{\psi_2}G_2 \leq \cdots \leq \widehat{\psi_M}G_M$.*
4. *Consider the residual capacity measured by the session s^i at the node v_j; then, for every session s^i, the normalized residual capacities $c_j^i = c_j - \sum_{k\neq i}\widehat{d_j^k}$ are decreasing with the order number of nodes, i.e., $c_1^i/\widehat{\psi_1} \geq c_2^i/\widehat{\psi_2} \geq \cdots \geq c_M^i/\widehat{\psi_M}$.*

A proof of Theorem 3 is provided in Appendix C.

Conclusion 1 in Theorem 3 implies that, for every session s^i, there exists some node v_{N^i}, such that $\widehat{d_{N^i}^i} > 0$ for all $j \leq N^i$ and $\widehat{d_{N^i}^i} = 0$ for all $j > N^i$. Conclusion 2 implies that for every node v_j, there exists some session s^{E_j}, such that $\widehat{d_j^{E_j}} > 0$ for all $i \leq E_j$ and $\widehat{d_j^{E_j}} > 0$ for all $i > E_j$. Let $V^i = \{v_1, v_2, \ldots, v_{N^i}\}$ denote the set of nodes to receive the flow from the session s^i, and $S_j = \{s^1, s^2, \ldots, s^{E_j}\}$ be the set of sessions to send the flow to the node v_j. Then, $V^{i+1} \subseteq V^i (1 \leq i \leq R)$ and $S_{j+1} \subseteq S_j (1 \leq j \leq M)$.

We now derive an explicit property of the session's equilibrium strategy $\widehat{d^i}$. It is a function of c_j^i and the incentive-compatible $\widehat{\psi}$. The Nash equilibrium $\widehat{d}$ of the SDG with the capacity configuration c and the incentive-compatible pricing vector $\widehat{\psi}$, which satisfies (4), is given by

$$\widehat{d_j^i} = \begin{cases} c_j^i - \left(\sum_{k=1}^{N^i} c_k^i - r^i\right)\dfrac{\sqrt{\widehat{\psi_j}c_j^i}}{\sum_{k=1}^{N^i}\sqrt{\widehat{\psi_k}c_k^i}}, & \text{if } j \leq N^i, \\ 0, & \text{if } j > N^i, \end{cases} \tag{8}$$

where, for every session s^i, the threshold N^i is determined by

$$A_{N^i}^i < r^i < A_{N^i+1}^i, \tag{9}$$

$$A_j^i = \sum_{k=1}^{j} c_k^i - \sqrt{c_j^i/\widehat{\psi_j}}\sum_{k=1}^{j}\sqrt{\widehat{\psi_k}c_k^i} \qquad j = 1, 2, \ldots, M, \tag{10}$$

$$A_{M+1}^i = \sum_{j=1}^{M} c_j^i = c - \sum_{l\neq i} r^l. \tag{11}$$

5 Incentive-Compatible Pricing Mechanism

For an AG with the capacity configuration c, we now derive the expression of an incentive-compatible pricing vector $\widehat{\psi}$. Without loss of generality, we assume that at the NEP $\widehat{d}$ all nodes receive stream; otherwise, we can prevent the session from sending stream to certain node by setting the pricing factors at sufficiently large values.

Let $\widehat{x_j} = c_j - \widehat{d_j}$ denote the residual capacity of the node v_j at the NEP and $\eta_j = (\widehat{x_j})^2/\widehat{\psi_j}$. As described in Section 2, if an incentive-compatible pricing vector $\widehat{\psi}$ exists, the corresponding Nash equilibrium $\widehat{d}$ has the properties described in Theorems 1 to 3. According to (8), for every session s^i, there exists a node v_{N^i} determined by (9), (10), and (11), such that $\widehat{d_j^i} > 0$ for $j \leq N^i$, and $\widehat{d_j^i} = 0$ for $j > N^i$. The equilibrium strategy of the session is described by (8). The following lemma shows that the residual capacity of the node v_j measured by any session (at the equilibrium) is proportional to η_j:

Lemma 1. *Under any incentive-compatible pricing vector $\widehat{\psi}$, the residual node capacities c_j^i at the corresponding Nash equilibrium are given by*

$$c_j^i = \frac{\eta_j}{\sum_{k=1}^{N^i}\eta_k}\left(\sum_{k=1}^{N^i}\widehat{x_k} + r^i\right) \qquad j = 1, 2, \ldots, M. \tag{12}$$

Then, we try to find the expression of an incentive-compatible pricing vector $\widehat{\psi}$. Any incentive-compatible pricing vector $\widehat{\psi}$ satisfies

$$\widehat{\psi_j} = \left(\frac{\widehat{x_j}}{\widehat{x_1}}\right)^2 \frac{\widehat{\psi_1}}{\varphi_j(\varphi_{j-1}+1)(\varphi_2+1)} \qquad j = 1, 2, \ldots, M, \tag{13}$$

where

$$\varphi_j = \frac{c_j^i + (N^i - 1)\widehat{x_j}}{\sum_{k=1}^{j-1} c_k + (N^i - 1)\sum_{k=1}^{j-1}\widehat{x_k} - \sum_{k:v_k\notin V^i} r^k}. \tag{14}$$

Since an incentive-compatible pricing vector $\widehat{\psi}$ can be specified to a multiplicative constant, in the rest of this paper, we assume that the node v_j is with the fixed pricing factor. Formulas (12) and (13) implies that if we knew a priori the set of nodes V^i over which each session s^i would send its flow under an incentive-compatible pricing vector ψ_i, then it could find the pricing factors using (12) and (13), where $j \geq 2$. The set V^i depends on the pricing vector ψ according to (9). Then, we derive a condition to determine N^i independent of ψ. To that end, let us define T^i_j for the session s^i:

$$T^i_j = \frac{\sum_{k=1}^{j} c_k - c_j/\left(\widehat{x_j}\sum_{k=1}^{j}\widehat{x_k}\right) - \sum_{m=j+1}^{R} r^M}{c_j/(\widehat{x_j}+i-1)}, \tag{15}$$

$$T^i_{M+1} = \sum_{i=1}^{R} r^i. \tag{16}$$

Since $c_j/\widehat{x_j} \geq c_{j+1}/\widehat{x_{j+1}}$, it is easy to verify that $T^i_j \leq T^i_{j+1}$. Thus, we have the following conclusion:

Let N^i be the set of nodes that the session s^i sends its flow (at the equilibrium) under an incentive-compatible pricing vector. Then,

$$T^i_{N^i} < r^i \leq T^i_{N^i+1}. \tag{17}$$

Note that condition (17) depends only on the NEP, the capacity configuration, and the throughput demands of the sessions. Since $T^i_j \leq T^i_{j+1}$ for all $j \in \{j|v_j \in V_i\}$, (17) determines N^i uniquely, for each session s^i. Given the set of nodes over which each session sends its flow, the incentive-compatible pricing factors are determined uniquely by (13).

The existence of an incentive-compatible pricing vector is presented as follows:

Theorem 4. *For the SDG, there exists a unique incentive-compatible pricing vector found by (13) and (14), where, for every session s^i, N^i is the node determined by (17). The corresponding Nash equilibrium $\widehat{d}$ is given by*

$$\widehat{d^i_j} = \begin{cases} \frac{(\widehat{x_j})^2/\widehat{\psi_j}}{\sum_{k=1}^{N^i}(\widehat{x_k})^2/\widehat{\psi_k}}\left(\sum_{k=1}^{N^i}\widehat{x_k} + r^i\right) - \widehat{x_j}, & j \leq N^i, \\ 0, & j > N^i. \end{cases} \tag{18}$$

Proof. Combined with (13), (14), and (8), (18) can be derived. □

6 Adaptive Pricing Algorithm

The analysis in the previous sections depends on the assumption that we have complete knowledge of the behavior of each session as well as its throughput demand. In practice, however, we might not be cognizant of the behavior of the sessions. Moreover, in large-scale P2P networks, it might be difficult to keep track of the individual throughput demands of a large number of sessions. Under such a scenario, an adaptive algorithm that determines the incentive-compatible pricing vector online would be more efficient.

6.1 APV Algorithm

In this section, we consider an APV algorithm. It updates the weighted vector based on the total flow currently offered to each node. The APV algorithm is described by the following iteration:

$$\psi_j(n+1) = \psi_j(n)e^{\rho_j\left(d_j(n)-d_j\right)} \qquad j = 1,\ldots,M, n \geq 1, \tag{19}$$

where ψ_j and $d_j(n)$ are the weighted factor and the observed flow of the node v_j at the nth iteration, and ρ_j is a positive constant. Note that the pricing factor of node is updated based only on information about the total flow offered to the node (local information). Thus, the APV algorithm is well suited for distributed implementation.

In fact, some previous works focus on the flow measurement of the node in P2P network. In [28], Saroiu et al. proposed a proactive crawling system to measure the node flow in Napster and Gnutella. Liang et al. [29] developed a powerful crawling system, which crawls the majority of the FastTrack Network's 20,000+ supernodes in less than 60 minutes. In [30], the authors provided an efficient approach for identifying the P2P application traffic through application level signatures. They identified the application level signatures by examining some available documentations and packet-level traces. And then, they utilized the identified signatures to develop online filters that can efficiently and accurately track the P2P traffic even on high-speed network links. In this paper, we use the proactive crawling system [28] to measure the traffic in a node.

6.2 The Convergence Analysis

The convergence analysis will be based on the assumption that the node flow configuration observed at the nth iteration is the one induced by $\psi(n)$. The assumption is satisfied, for example, if the APV algorithm determines the update $\psi(n+1)$ only after the SDG under the pricing vector $\psi(n+1)$ converges to its unique Nash equilibrium. Let $D(\psi)$ denote the node flow configuration under the pricing vector ψ. We are now ready to state formally the assumptions for the convergence analysis.

Assumption G:

- G.1. For any $n \geq 1$, $d(n) = D(\psi)$.
- G.2. If D_j is a Lipschitz continuous function of ψ_j, then there exists B_j $(0 < B_j < \infty)$, such that $|D_j(\psi) - D_j(\psi + \Delta\psi_j e_j)| \leq B_j|\Delta\psi_j|$ for any $j = 1,\ldots,M$,

where $e_j = (1,0,0,\ldots,0)$ is the vector in R^j.

The Lipschitz continuity assumption essentially means that small changes in the pricing vector cannot induce drastic changes in the link flow configuration. If $D(\psi)$ is differentiable, G.2 implies that its partial derivatives are absolutely bounded.

Theorem 5. *Consider the SDG that satisfies Assumptions G. If*

$$\rho_j B_j < 1 \qquad j \in \{j|v_j \in V\}, \tag{20}$$

the APV algorithm described by (19) converges to the incentive-compatible pricing vector. Moreover, the node flow configuration converges to the NEP $\widehat{d}$, i.e., $\lim_{n\to\infty} d(n) = \widehat{d}$.

A proof of Theorem 5 is provided in Appendix D.

This result implies that we can run the NEP for a wide range of sessions whose aggregate behavior satisfies Assumptions G, without having explicit knowledge of their individual cost functions or throughput demands.

In summary, the APV algorithm exhibits the following properties:

- Robustness: With a high probability, the APV algorithm works well even in the face of high traffic. It is because that the pricing factor of a node is updated based only on information about the total flow offered to the node.
- Scalability: The APV algorithm can handle the system generated by millions of end hosts and applications.
- Efficiency: The APV algorithm can compute the pricing vector matching the incentive-compatible pricing vector.

7 Simulations

In this section, we simulate our proposed mechanism and algorithms to investigate their performances.

7.1 Simulation Methodology

We use Matlab for simulation. In this section, we first present the simulation environment including network topology, content distribution, and peer characteristics, and then describe the performance metric we used.

Network topology. The basic network topology is a euclidean space model. A euclidean model is a hypercubic in a D-dimensional euclidean space. Nodes are randomly located in the hypercubic. The distance metric corresponds to the euclidean distance metric between the nodes. Using this model, we can generate topology maps with easy control of parameters. We apply the APV algorithm by randomly selecting groups in a hypercubic in a 2D euclidean space. The edge longitudes of the hypercubic are fixed to 3,000 ms (for a given link speed) in all the experiments. After the overlay is formed, 200 groups are generated accordingly.

The lifetime of each session is exponentially distributed with an average length of 10 minutes. In the simulation, a number of peers periodically join a session at the same time according to a Poisson arrival process. It will induce a flash crowd. No peer leaves the session during the flash crowding.

Content distribution. A combination of exponential on-off and Pareto on-off traffic sources are used in the simulation as the media peers. Unless otherwise specified, the shape parameter for Pareto sources is set to 1.5 and the on-time and off-time are both set to 0.5 second. The traffic conditioners are configured with one profile for each traffic source, with peak rate and bucket size set to the on-off source peak rate and the maximum amount of traffic sent during an on-time, respectively.

MPEG-4 layered scalable Progressive Fine Granularity Scalable (PFGS) [19] format video clips are chosen as multimedia contents in our simulation. We assume there are some video clips in our multimedia distribution system. Each video clip is encoded in 1.28 Mbps and the length of the video clips follows a normal distribution in the range of 3 to 5 minutes, correspondingly to 37.8 to 48 Mbytes in file sizes, respectively.

System load. We also characterize the system load by load index. The load index is defined as d/c, where d is the current system load and c is the current system capacity. To measure the performance of the peer, we have primarily used the Autobench [4], which allows us to generate heavy workloads (i.e., high load index) and simulate simultaneous connections from large client populations. To observe the impact of our mechanism during the flash crowd, we have used a modified version of the Siege application [21].

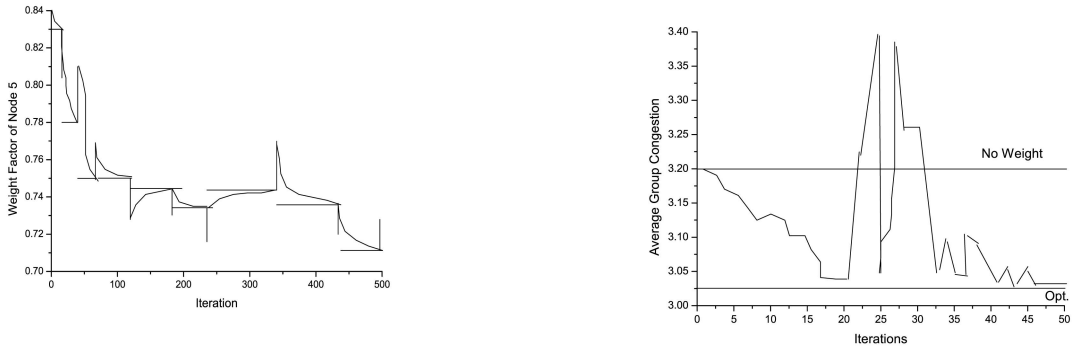
Peer storage capacity and reliability. Each peer contributes some resource to the P2P-based multimedia distribution service. Suppose the storage contributed by a peer follows a normal distribution in the range of 300 Mbytes and 2 Gbytes, which approximately supports 8 to 50 video clips. Considering the storage capacity of the current personal computer used in a home or office environment, a 500-Mbyte-storage donation is reasonable and acceptable.

Another important characteristic is the reliability of the peer. We assume that peer reliability of sustaining the service follows a normal distribution in the range of 0.1 to 0.9.

7.2 Performance Metrics

We use a number of engineering and economic metrics to evaluate our experiments. The engineering metrics include the average delay and the average packet loss rate. The economic performance metrics include media quality (video quality) and the quality of a session. The averages are computed as exponentially weighted moving averages.

- *Average latency* represents the average access time between the requestor peer and the media peer. We assume a virtual time to reflect the performance perceived by the end peer where the time may be Round Trip Time (RTT), hops, or even certain economic cost of the path between two peers. We choose RTT as the distance metric in our simulation.
- *Average packet loss rate* represents the average packet loss rate at the nodes with high traffic.
- *The average probability of user request blocking* represents the average probability that the requests of the requestor peers are blocked.
- *Video quality* represents the video quality *perceived by the peers*. We use peak signal-to-noise ratio ($PSNR$) as the metric to measure video quality. For an 8-bit image with the intensity values between 0 and 255, the $PSNR$ is defined as $PSNR = 20 \log_{10} 255/RMSE$, where $RMSE$ stands for root mean squared error. Given an original image f and the compressed or degraded image f', the $RMSE$ can be calculated as follows [2]: $RMSE = \sqrt{M \sum_{x=0}^{N} \sum_{y=0}^{M} [f(x,y) - f'(x,y)]^2 / N}$. In our simulation, the perceived video quality is affected by both the available bandwidth and packet loss ratio in the network link between requestor and content provider. Generally, the closer the requestors to service nodes are, the higher bandwidth and lower packet loss ratio the node tends to have; therefore, the better video quality is perceived by the requestor. The simplest streaming protocols, RTP/UDP/IP, are used in our simulation to study the video quality in different multimedia distribution schemes, even though some other advanced techniques, such as rate control, error control, and error concealment, are very helpful to enhance the end-to-end streaming performance.



(a) (b)

Fig. 4. The performance of APV algorithm.

- *Average quality of the media sessions* quantify the performance of a media streaming system. *Unlike the video quality, it perceived by the media sessions*. The quality of a media session is defined by $Q = (\sum_{k=1}^{T} Z(k))/T$ [31], where T is the total number of packet in a media session, and $Z(k)$ is a variable that is set to 1 if the packet k arrives at the receiver before its scheduled play-out time, and 0 otherwise. Here, T represents the granularity of the duration to measure the QoS of a session. The quality is different from throughput because it considers the deadline of each packet. The parameter Q captures other performance parameters such as packet delay, packet loss, and jitter. A packet that misses its deadline is discarded and, therefore, does not contribute to the quality of a media session, similar to a lost packet.

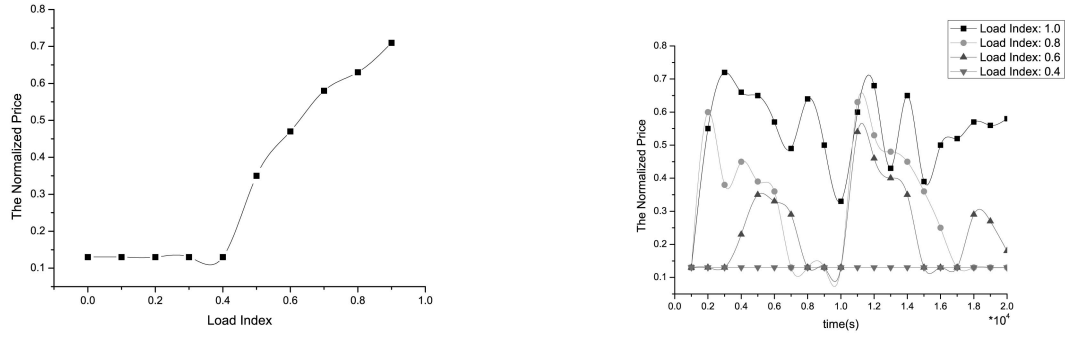
7.3 The Performance of APV Algorithm

To investigate the behavior of the APV algorithm, we randomly select a group with the capacity configuration $c = (15, 12, 10, 8, 6)$. We start with two active sessions sharing the group. When the APV algorithm determined the maximally efficient pricing vector, we add another media session.

Fig. 4a shows the relationship between the pricing factor of a random node v and the number of iterations of the adaptive scheme. The spikes in the plot correspond to points where a new session joins the SDG. The horizontal line segments indicate the value of the maximally efficient pricing factor that is determined analytically. The length of each line segment indicates the number of iterations required for convergence. As shown in Fig. 4a, the APV algorithm determined successfully the maximally efficient pricing factor. When the number of active sessions ranges from 2 to 5, the number of iterations required for convergence (tolerance 10^{-5}) is 16, 24, 28, 39, and 51, respectively. As already explained, the adaptive scheme updates the pricing vector periodically (every 3s), without waiting for the SDG to converge at each iteration. Convergence to the maximally efficient pricing vector slows down as the number of sessions increases. This is because: 1) convergence of the SDG itself (for the fixed ψ) slows down as the number of session increases and 2) the constant ρ_j to determine the rate of change decreases with the number of sessions. Fig. 4b shows the relationship between the average group congestion and the number of iterations. The horizontal lines in the figure indicate the average group congestion at the network optimum and for the case with no pricing ($\psi_j = 1$). From Fig. 4b, we observe that the APV algorithm closely approximates the optimum and it can efficiently avoid the hotspot.

We compare the performance of APV algorithm as the load index increases. Fig. 5a shows that the price of a random node v (the price determined by (3)) increases due to the increasing congestion level as the load index exceeds 0.4. In response, the traffic of the random node v backs off. Fig. 5b shows that the dynamic variation of the price at four different load indices confirms this trend.

The results in this section indicate that the APV algorithm is stable and effective.



(a) (b)

Fig. 5. System dynamics under APV algorithm with increase in load index.

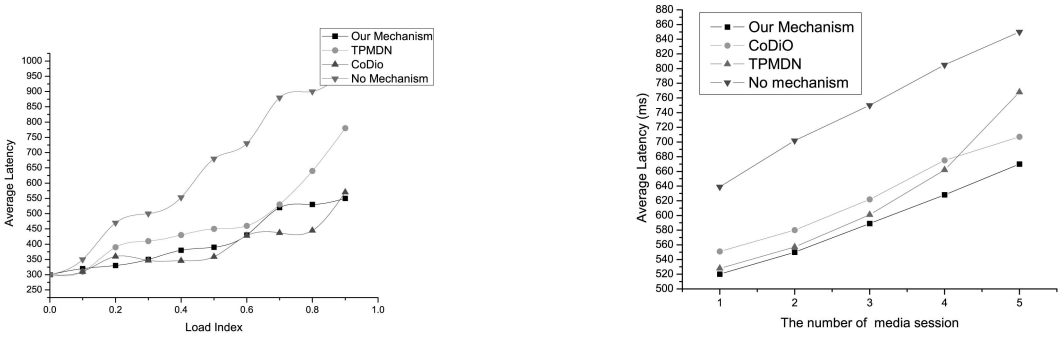


Fig. 6. Average latency comparison of different mechanisms. (a) Single-source streaming distribution. (b) Multisource streaming distribution.

7.4 Performance Comparison

In order to evaluate the effectiveness of our proposed incentive-compatible mechanism for hotspot avoidance in a P2P-based multimedia distribution service, we compare our mechanism with three mechanisms from the viewpoint of single-source and multisource scenario streaming distributions. In the single-source scenario, the number of concurrent media sessions R is 1. In the multisource scenario, the value of R is greater than 1. The three mechanisms are listed as follows:

- TPMDN mechanism: It provides two heuristic replication strategies, intergroup replication, and intragroup replication to avoid the routing hotspots [2].
- CoDiO mechanism: It introduces the concept of congestion-distortion optimized (CoDiO) packet scheduling. The mechanism preencodes streams at the source node and varies the rate of streams when the high traffic happens [31].
- No mechanism: It represents the case when no avoiding mechanism is running.

7.4.1 Average Latency

We use average latency to study the performance of four mechanisms.

We first consider the single-source streaming distribution $(R = 1)$ by varying the value of the load index. Fig. 6a shows that when the load index increases, the average latency increases accordingly. Our mechanism is close to CoDiO. However, the latency of our mechanism is less than that of TPMDN mechanism and the case when no avoiding mechanism is done. When the network is idle, the latency is almost the same for the different mechanisms. At the peak of the flash crowd $(\text{load index} = 0.9)$, the average latency of our mechanism is 550 ms, which demonstrates that our mechanism is efficient for the single-source distribution.

Second, we consider the multisource streaming distribution $(R = 1)$ by varying the number of concurrent media sessions from 1 to 5. As illustrated in Fig. 6b, with the increase of the number of the media sessions, the average latency steadily increases accordingly. Under a fixed storage capacity (800 Mbytes), our mechanism outperforms the other three mechanisms. The latency in our proposed mechanism is 5 percent less than that of the TPMDN mechanism and CoDiO, and 30 percent less than the case when no avoiding mechanism is done. For $R = 5$, the latency of our mechanism is 650 ms, whereas other mechanisms reach 700 ms at least. This is mainly because the APV algorithm in our mechanism can be converged fast.

7.4.2 Average Packet Loss Ratio

We use average packet loss ratio to study the performance of four mechanisms.

In the single-source streaming distribution $(R = 1)$, we vary the value of load index from 0.1 to 0.9. As shown in Fig. 7a, with the increase of the value of the load index, the average packet loss ratio increases accordingly. The CoDiO

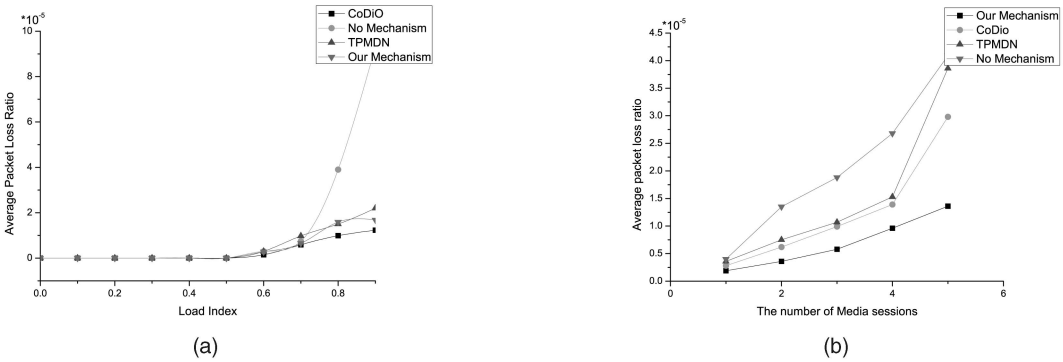


Fig. 7. Average packet loss ratio comparison of different mechanisms. (a) Single-source streaming distribution. (b) Multisource streaming distribution.

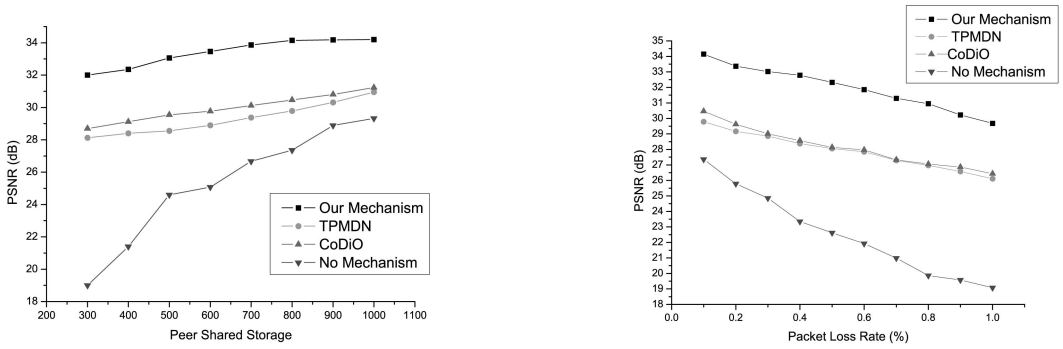


Fig. 8. Video quality comparison of different mechanisms for the single-source distribution. (a) Varying the peer storage size. (b) Varying the average packet loss ratio.

mechanism has the best performance about the average packet loss ratio. Our mechanism without rate distortion is close to CoDiO and outperforms the TPMDN mechanism and no mechanism. This is because the CoDiO adopts the rate distortion and thus has the best performance.

Now, we consider the multisource streaming distribution $(R = 1)$ by varying the number of concurrent media sessions from 1 to 5. Fig. 7b shows that with the increase of K, the network experiences at least 33 percent packet loss for other mechanisms. With our mechanism, the loss ratio is only 10 percent for the same number of sessions. The loss ratio curve follows the same pattern as that of the rate curve, i.e., our mechanism is more useful than other mechanisms when the network is not idle or heavily congested.

7.4.3 Video Quality

We then use video quality as the metric to study the performance of four mechanisms.

First, we consider the single-source streaming distribution. We fix the average packet loss ratio as 0.9 to simulate the flash crowding event and vary the peer storage size. Fig. 8a shows the $PSNR$ of four mechanisms by varying the peer storage size from 300 Mbytes to 1 Gbyte. Generally, the increase of peer storage will in turn increase the perceived video quality when all four mechanisms are applied. From Fig. 8a, we can see that the average $PSNR(\mathrm{s})$ using our mechanism, TPMDN, CoDiO, and no mechanism are 33.46, 29.3, 29.97, and 25.29 dB, respectively. In our mechanism, when peer storage is increased from 300 Mbytes to 1 Gbyte, there is a significant improvement for system performance, the $PSNR$ increases from 32 to 34.2 dB. Then, we fix the peer storage capacity as 800 Mbytes and vary the average packet loss ratio. The results also imply that for single-source media distribution system with our mechanism, the video quality is still good at the peak of the flash crowd. As shown in Fig. 8b, perceived video quality in all mechanisms decreases as the packet loss ratio increases. However, our mechanism always performs the best in all four mechanisms. When the average packet loss ratio is about 0.5 percent, the $PSNR$ of our mechanism is about 31.97 dB, while the $PSNR$ of TPMDN is 27.79 dB and the $PSNR$ of CoDiO is about 28.14 dB. Furthermore, for no avoiding mechanism, the $PSNR$ is only 22.55 dB. The results imply that for single-source media distribution system with our mechanism, video quality is good whenever the network is idle or heavily congested.

Second, we consider the multisource streaming distribution. We fix the number of concurrent media sessions as $R = 5$ and vary the peer storage size. When $R = 5$, the network should be heavily congested. As plotted in Fig. 9a, with the increase of the peer storage size, the perceived video quality increases accordingly. In fact, the increase of peer shared storage help in the hotspot avoidance

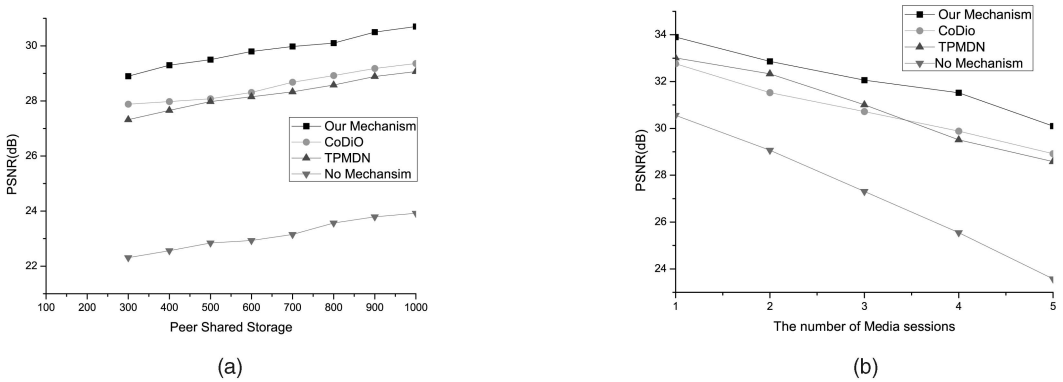


Fig. 9. Video quality comparison of different mechanisms for the multisource distribution. (a) Varying the peer storage size. (b) Varying the number of media sessions.

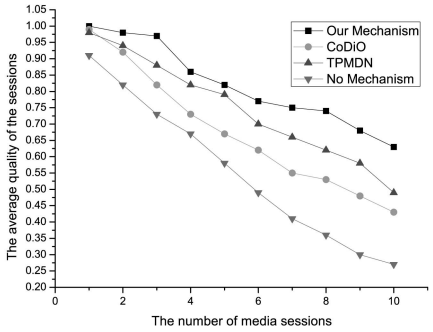


Fig. 10. The average quality of the sessions of different mechanisms, varying the number of media sessions.

because the systems have more caches for the media distribution. However, we only consider the impact of different mechanisms in the same share space. We can see that the average $PSNR(\mathrm{s})$ using our mechanism, CoDiO, TPMDN, and no mechanism are 29.55, 28.11, 27.97, and 22.95 dB, respectively. When peer storage is increased from 300 Mbytes to 1 Gbyte, there is a significant improvement for system performance with our mechanism from 28.9 to 30.7 dB. The results also imply that multisource distribution system with our mechanism still works well at the peak of the flash crowd. Then, we fix the peer storage capacity as 800 Mbytes and vary the number of concurrent media sessions from 1 to 5. Fig. 9b shows that the video quality in our proposed architecture is 10 percent better than that of the TPMDN and the CoDiO, and 30 percent better than the case when no avoiding mechanism is done. The results imply that our mechanism still outperforms the other mechanisms whenever the network is idle or heavily congested.

7.4.4 The Average Quality of the Sessions

Now, we study the quality from the viewpoint of the media sessions by using the metric defined in Section 7.2. Here, packets that miss their play-out deadlines are considered as lost.

Fig. 10 shows that the average quality of the sessions is close to 1 for less than three concurrent sessions if our mechanism is used. Otherwise, the other mechanisms cannot maintain the quality to 1 for the same number of concurrent sessions. If the network load is increased, the session quality deteriorates with or without our mechanism. For higher number of sessions, the quality is low for each mechanism because the network is extremely congested. However, our mechanism improves the system performance more than 20 percent than other mechanisms in such a condition. Thus, our mechanism is necessary and efficient when the network is heavily congested. The reason is given as follows: With our incentive mechanism, the load is distributed among all peers in the same session because peers act as a supplier when necessary. However, it cannot be achieved without our mechanism.

8 Conclusion

We have developed a methodology to overcome the flash crowd (routing hotspot) problem for both single-source and multisource P2P streaming applications by means of appropriate pricing mechanism. Assuming that the cost of shipping a unit of flow over any node is proportional to the congestion level at the node, we showed that the nodes can always determine its pricing factors. The factors provide incentives to the sessions to implement a Nash equilibrium. The incentive-compatible pricing vector was shown to be unique and its expression was specified explicitly. We introduced the APV algorithm for the computation of the incentive-compatible pricing vector. The APV algorithm is distributed in the sense that the pricing factor of each node is updated based only on the difference between the total flow currently received by the node and the NEP. The simulation results show that hotspots can effectively be avoided at low cost.

In the future, we need to explore the above-proposed mechanism further. For the different applications, it is necessary to study the detail versions of the mechanism and determine whether such modifications can still be efficient to avoid the routing hotspot.

We should note that it is difficult for the media sessions to reach the NEP as the NEP is too sensitive. This motivates the question of protocol equilibrium: Can we design the application-level protocols that lead to efficient network operation for avoiding the routing hotspot?

Appendix A

Proof of Theorem 1

We define the point-to-set mapping $d \in D \rightarrow \Gamma(d) \subset D$, where $\Gamma(d) = \{\widehat{d} \in D : \widehat{d}^i \in \arg\min_{g^i \in D^i} J^i(d^1, \ldots, g^i, \ldots, d^R)\}$.

Γ is an upper semicontinuous mapping (according to the property P2). Γ maps each point of the convex compact set D into a closed (according to P2) convex (according to P3) subset of D. According to the Kakutani fixed point theorem, there must exist a fixed point $d \in \Gamma(d)$, and such a point easily reaches a Nash equilibrium. □

Appendix B

Proof of Theorem 2

Let $\widetilde{d}$, $\widehat{d} \in D$ be two NEPs. Thus, $\widetilde{d}$ and $\widehat{d}$ satisfy the KKT conditions (5) and (6), i.e.,

$$K_j^i\left(\widetilde{d}_j^i, \widetilde{d}_j\right) = \widetilde{\lambda}^i; K_j^i\left(\widetilde{d}_j^i, \widetilde{d}_j\right) \geq \widetilde{\lambda}^i \quad \text{if } \widetilde{d}_j^i = 0, \tag{21}$$

$$K_j^i\left(\widehat{d}_j^i, \widehat{d}_j\right) = \widehat{\lambda}^i; K_j^i\left(\widehat{d}_j^i, \widehat{d}_j\right) \geq \widehat{\lambda}^i \quad \text{if } \widehat{d}_j^i = 0. \tag{22}$$

First, we prove that $\widetilde{d} = \widehat{d}$ for every node $v_j \in V$. Assume that $\widetilde{\lambda}^i \geq \widehat{\lambda}^i$ and $\widetilde{d}_j \leq \widehat{d}_j$ for the node v_j and the session s^j. If $\widehat{d}_j^i = 0$, then we have

$$\widehat{d}_j^i \leq \widetilde{d}_j^i. \tag{23}$$

If $\widehat{d}_j^i > 0$, then (21) and (22) together with the properties P1 to P5 imply that

$$K_j^i\left(\widehat{d}_j^i, \widehat{d}_j\right) = \widehat{\lambda}^i \leq \widetilde{\lambda}^i \leq K_j^i\left(\widetilde{d}_j^i, \widetilde{d}_j\right) \leq K_j^i\left(\widetilde{d}_j^i, \widehat{d}_j\right), \tag{24}$$

where inequality (24) follows the monotonicity of K_j^i in its second argument d_j. Now, since K_j^i is nondecreasing in its

first argument d_j^i, this implies that when $\widehat{d}_j^i > 0$, (23) is satisfied. Thus, we have the following conclusion C1: When $\widetilde{\lambda}^i \geq \widehat{\lambda}^i$ and $\widetilde{d}_j \leq \widehat{d}_j$, $\widehat{d}_j^i \leq \widetilde{d}_j^i$ is satisfied. Similarly, we have the following symmetric conclusion C2: When $\widetilde{\lambda}^i \leq \widehat{\lambda}^i$ and $\widetilde{d}_j \geq \widehat{d}_j$, $\widehat{d}_j^i \geq \widetilde{d}_j^i$ is satisfied.

Let $V_1 = \{v_j : \widehat{d}_j > \widetilde{d}_j\}$ and $V_2 = V - V_1 = \{v_j : \widehat{d}_j \leq \widetilde{d}_j\}$. Assume that $S^1 = \{s_i : \widehat{\lambda}^i < \widetilde{\lambda}^i\}$. We also assume that V_1 is nonempty. Due to conclusion C2 and $\sum_{v_i \in V} \widehat{d}_j^i = \sum_{v_i \in V} \widetilde{d}_j^i = r^i$, for any session $s^i \in S_1$, we have

$$\sum_{v_j \in V_1} \widehat{d}_j^i = r^i - \sum_{v_j \in V_2} \widehat{d}_j^i \leq r^i - \sum_{v_j \in V_2} \widetilde{d}_j^i = \sum_{v_j \in V_1} \widetilde{d}_j^i. \quad (25)$$

Note that conclusion C1 implies that $\widehat{d}_j^i \leq \widetilde{d}_j^i$ for any $v_j \in V_1$ and $s^i \notin S_1$, it follows that

$$\sum_{v_j \in V_1} \widehat{d}_j = \sum_{v_j \in V_1} \sum_{s_i \in S_1} \widehat{d}_j^i \leq \sum_{v_j \in V_1} \sum_{s_i \in S_1} \widetilde{d}_j^i = \sum_{v_j \in V_1} \widetilde{d}_j. \quad (26)$$

This inequality implies that V_1 is an empty set. It obviously contradicts our definition of V_1. By symmetry, we can conclude that the set $\{v_j : \widehat{d}_j < \widetilde{d}_j\}$ is empty. Thus, it is correct that

$$\widetilde{d}_j = \widehat{d}_j, \quad \text{for any } v_j \in V. \quad (27)$$

We now prove that $\widehat{\lambda}^i = \widetilde{\lambda}^i$ for each session s^i. To this end, note that (21) can be strengthened as follows: $\widehat{\lambda}^i < \widetilde{\lambda}^i$ and $\widehat{d}_j = \widetilde{d}_j$ implies that either $\widehat{d}_j^i < \widetilde{d}_j^i$ or $\widehat{d}_j^i = \widetilde{d}_j^i = 0$.

Indeed, if $\widehat{d}_j^i = 0$, then the implication is trivial. If $\widehat{d}_j^i > 0$, similarly to (23), we have that $K_j^i(\widehat{d}_j^i, \widehat{d}_j) < K_j^i(\widetilde{d}_j^i, \widehat{d}_j)$ if $\widehat{d}_j^i < \widetilde{d}_j^i$.

Assume that $\widehat{\lambda}^i < \widetilde{\lambda}^i$ for session $s^i \in S$. Since $\sum_{v_j \in V} \widehat{d}_j^i = r^i > 0$, then $\widehat{d}_j^i > 0$ for at least one node v_j. Thus, we have $\sum_{v_j \in V} \widetilde{d}_j^i > \sum_{v_j \in V} \widehat{d}_j^i = r^i$.

It contradicts the demand constraint for session s^i. We therefore conclude that $\widehat{\lambda}^i < \widetilde{\lambda}^i$ does not hold for any session s^i. A symmetric argument may be used to show that $\widehat{\lambda}^i > \widetilde{\lambda}^i$. Thus, $\widehat{\lambda}^i = \widetilde{\lambda}^i$ for any session s^i. Combined with (27), the uniqueness of the NEP is thus proved. □

APPENDIX C
PROOF OF THEOREM 3

Since the cost function J_j^i is a positive, convex increasing function and the condition $c_j < \widehat{d}_j$ is satisfied according to the KKT condition (recall that the latter are strictly increasing in their first argument), we have $0 \leq K_1^i \leq K_2^i \leq \cdots \leq K_M^i$. Choose an arbitrary session s^i. The claim holds trivially for $d_j^i = 0$. Assume, then, that $d_j^i > 0$. From the KKT conditions, we have that $K_j^i(d_j^i, d_j) \leq K_{\widehat{j}}^i(d_{\widehat{j}}^i, d_{\widehat{j}})$.

For $d_j^i > d_{\widehat{j}}^i$, it implies that $d_{\widehat{j}}^i > 0$. We have $K_j^i(d_j^i, d_j) \geq K_{\widehat{j}}^i(d_{\widehat{j}}^i, d_{\widehat{j}})$.

Thus, we have $K_j^i(d_j^i, d_j) \leq K_{\widehat{j}}^i(d_{\widehat{j}}^j, d_{\widehat{j}}) < K_{\widehat{j}}^i(d_{\widehat{j}}^i, d_{\widehat{j}})$. Due to condition (4), we have $d_1^i/\psi_1 \geq d_2^i/\psi_i \geq \cdots \geq d_M^i/\psi_M$. In other words, Conclusion 3.1 is true.

Using the same method, we can prove that Conclusions 3.2, 3.3, and 3.4 are true. □

APPENDIX D
PROOF OF THEOREM 5

Without loss of generality, we assume that sessions s^1 and s^2 share the same AG. Let d^1 and d^2 be the optimal feasible flows of the session s^1 and s^2, respectively.

We first establish that each component of $d(n)$ increases or decreases monotonically with n. Let v_j be a node for which $d_j^1(1) < d_j^1(1)$. Noting that, by its definition, $d^2(2)$ is optimal for session s^2 against $d^2(2)$, and that, similarly, $d^2(4)$ is optimal against $d^1(3)$, it follows from Theorem 3 that $d_j^2(2) \geq d_j^2(4)$. A symmetric argument can now be employed to establish that $d_j^1(3) \leq d_j^1(5)$. Proceeding inductively and recalling that the flows of the session s^1 remain fixed at each even step, it follows for each odd n: $d_j^1(n) = d_j^1(n+1) \leq d_j^1(n+2)$.

Similarly, for each even n : $d_j^2(n) = d_j^2(n+1) \geq d_j^2(n+2)$.

Since the flows are bounded, this implies that $d_j^1(n)$ and $d_j^2(n)$ converge as $n \to \infty$. Since the sum of flows on both nodes is constant, this obviously implies similar convergence for the flows on the second node, so that $d(n)$ converges to some flow vector $d = (d^1, d^2)$.

Due to the continuity of the cost functions, it follows that d^1 is optimal for session s^1 against d^2 and d^2 is optimal for session s^1 against d^2, so that d is the NEP. □

ACKNOWLEDGMENTS

The authors would like to thank Professor Charles Lin of the University of Western Ontario for valuable suggestions that improved this paper. The short version of this paper was presented at the IEEE International Conference on Multimedia and Expo (ICME 2007). The work reported in this paper was supported by the NSFC under Grant 90612013, the National High Technology Research and Development Program of China under Grant 2006AA01Z304, the 111 Project under B08004, and the NCET of MOE, China.

REFERENCES

[1] D.A. Tran et al., "Scalable Media Streaming in Large P2P Networks," *Proc. ACM Int'l Conf. Multimedia (Multimedia '02)*, pp. 247-256, Dec. 2002.
[2] Z. Xiang et al., "Peer-to-Peer Based Multimedia Distribution Service," *IEEE Trans. Multimedia*, vol. 6, pp. 343-355, Apr. 2004.
[3] D.A. Tran et al., "A Peer-to-Peer Architecture for Media Streaming," *IEEE J. Selected Areas Comm.*, vol. 22, pp. 121-133, Jan. 2004.
[4] Z. Liu et al., "On Peer-to-Peer Multimedia Content Access and Distribution," *Proc. IEEE Int'l Conf. Multimedia and Expo (ICME '06)*, pp. 557-560, 2006.
[5] S. Sen et al., "Analyzing Peer-to-Peer Traffic across Large Networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 2, pp. 219-232, Apr. 2004.
[6] S. Androutsellis-Theotokis et al., "A Survey of Peer-to-Peer Content Distribution Technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335-371, Dec. 2004.
[7] C. Wu et al., "Optimal Peer Selection for Minimum-Delay Peer-to-Peer Streaming with Rateless Codes," *Proc. ACM Workshop Advances in Peer-to-Peer Multimedia Streaming (P2PMMS '05)*, pp. 69-78, Nov. 2005.

[8] R.T.B. Ma et al., "Incentive and Service Differentiation in P2P Networks: A Game Theoretic Approach," *IEEE/ACM Trans. Networking,* vol. 14, no. 5, pp. 978-991, Oct. 2006.
[9] S. Zhong et al., "Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad Hoc Networks," *Proc. IEEE INFOCOM '03,* pp. 1987-1997, 2003.
[10] A.C. Fuqua et al., "Economic Behavior of Peer-to-Peer Storage Networks," *Proc. Second Workshop Economics of Peer-to-Peer Systems (P2P Econ '03),* June 2003.
[11] D. Rubenstein et al., "Can Unstructured P2P Protocols Survive Flash Crowds?" *IEEE/ACM Trans. Networking,* vol. 13, no. 3, pp. 501-512, June 2005.
[12] D. Fudenberg and J. Tirole, *Game Theory.* MIT Press, 1991.
[13] B. Tuffin, "Charging the Internet without Bandwidth Reservation: An Overview and Bibliography of Mathematical Approaches," *J. Information Science and Eng.,* vol. 19, no. 5, pp. 765-786, Sept. 2003.
[14] L.A. DaSilva, "Pricing for QoS-Enabled Networks: A Survey," *IEEE Comm. Surveys,* vol. 3, no. 2, pp. 2-8, 2000.
[15] J. Altmann, B. Rupp, and P. Varaiya, "Internet Demand under Different Pricing Schemes," *Proc. First Ann. ACM Conf. Electronic Commerce (EC '99),* pp. 9-14, Nov. 1999.
[16] J. Shu and P. Varaiya, "Pricing Network Services," *Proc. IEEE INFOCOM '03,* vol. 2, pp. 1221-1230, 2003.
[17] P. Marbach, "Priority Service and Max-Min Fairness," *Proc. IEEE INFOCOM '02,* vol. 1, pp. 266-275, 2002.
[18] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan, "Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability," *J. Operational Research Soc.,* vol. 49, pp. 237-252, 1998.
[19] J.K. MacKie-Mason and H.R. Varian, "Pricing the Internet," *Public Access to the Internet,* B. Kahin and J. Keller, eds., pp. 269-314, MIT Press, 1995.
[20] P. Reichl, G. Fankhauser, and B. Stiller, "Auction Models for Multiprovider Internet Connections," *Proc. Messung, Modelierung und Bewertung (MMB '99),* pp. 71-75, Sept. 1999.
[21] N. Semret and A.A. Lazar, "Design, Analysis and Simulation of the Progressive Second Price Auction for Network Bandwidth Sharing," Technical Report CU/CTR/TR 487-98-21, Columbia Univ., Apr. 1998.
[22] P. Maillé and B. Tuffin, "Multi-Bid Auctions for Bandwidth Allocation in Communication Networks," *Proc. IEEE INFOCOM '04,* Mar. 2004.
[23] N. Semret, "Market Mechanisms for Network Resource Sharing," PhD dissertation, Center for Telecomm. Research, Columbia Univ., 1999.
[24] H. Chen and Y. Li, "Intelligent Flow Control under Game Theoretic Framework," *Telecommunications Optimization: Heuristic and Adaptive Techniques,* D.W. Come, G.D. Smith, and M.J. Oats, eds., Wiley, 2000.
[25] R.J. Gibbens and F.P. Kelly, "Distributed Connection Acceptance Control for a Connectionless Network," *Teletraffic Engineering in a Competitive World,* P. Key and D. Smith, eds., pp. 941-952, Elsevier, 1999.
[26] R. Kelly, "The Progressive Second Price Mechanism in a Stochastic Environment," *Netnomics,* vol. 5, no. 2, pp. 119-147, Nov. 2003.
[27] R. Kelly, "Multi-Bid versus Progressive Second Price Auctions in a Stochastic Environment," *Proc. Fourth Int'l Workshop Internet Charging and QoS Technology (ICQT '04),* pp. 318-327, Oct. 2004.
[28] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking (MMCN '02),* pp. 156-170, 2002.
[29] J. Liang, R. Kumar, Y. Xi, and K. Ross, "Pollution in P2P File Sharing Systems," *Proc. IEEE INFOCOM,* 2005.
[30] S. Sen, O. Spatscheck, and D. Wang, "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures," *Proc. 13th Int'l Conf. World Wide Web (WWW),* 2004.
[31] E. Setton, J. Noh, and B. Girod, "Rate-Distortion Optimized Video Peer-to-Peer Multicast Streaming," *Proc. ACM Workshop Advances in Peer-to-Peer Multimedia Streaming (P2PMMS '05),* pp. 39-48, Nov. 2005.
[32] M.J. Osborne and A. Rubenstein, *A Course in Game Theory.* MIT Press, 1994.
[33] C.H. Papadimitrou, "Algorithms, Games, and the Internet," *Proc. 33rd Ann. ACM Symp. Theory of Computing (STOC),* 2001.
[34] *Akamai,* http: //www.akamai.com, 2008.
[35] V. Pai, P. Druschel, and W. Zwaenepoel, "Flash: An Efficient and Portableweb Server," *Proc. USENIX Ann. Technical Conf. (USENIX '99),* June 1999.
[36] M. Welsh, D. Culler, and E. Brewer, "SEDA: An Architecture for Well-Conditioned, Scalable Internet Services," *Proc. 18th ACM Symp. Operating System Principles (SOSP '01),* Oct. 2001.
[37] J. Hu and D. Schmidt, "JAWS: A Framework for High Performance Web Servers," *Domain-Specific Application Frameworks: Frameworks Experience by Industry,* M. Fayad and R. Johnson, eds., John-Wiley, 1999.
[38] *Thttpd,* http://www.acme.com/software/thttpd/, 2008.
[39] X. Chen and J. Heidemann, "Experimental Evaluation of an Adaptive Flash Crowd Protection System," Technical Report ISI-TR-2003-573, Univ. of Southern California/Information Sciences Inst., July 2003.
[40] Google, http://www.google.com, 2008.
[41] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites," *Proc. 11th Int'l Conf. World Wide Web (WWW '02),* May 2002.
[42] W. Zhao and H. Schulzrinne, "Predicting the Upper Bound of Web Traffic Volume Using a Multiple Time Scale Approach," *Proc. 12th Int'l Conf. World Wide Web (WWW '03),* May 2003.
[43] S. Iyer, A. Rowstron, and P. Druschel, "SQUIRREL: A Decentralized, Peer-to-Peer Web Cache," *Proc. 21st ACM Symp. Principles of Distributed Computing (PODC '02),* July 2002.
[44] A. Stavrou, D. Rubenstein, and S. Sahu, "A Lightweight, Robust P2P System to Handle Flash Crowds," *Proc. 10th IEEE Int'l Conf. Network Protocols (ICNP '02),* Nov. 2002.

Zhen Yang received the BS degree in mathematics from Xiangtan University in 1997 and the PhD degree from the Beijing University of Posts and Telecommunications, China, in 2007. He is a lecturer in the Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, School of Computer, Beijing University of Posts and Telecommunications, China. His current research focuses on multimedia systems and networking.

Huadong Ma received the BS degree in mathematics from Henan Normal University in 1984, the MS degree in computer science from the Shenyang Institute of Computing Technology, Chinese Academy of Science, in 1990, and the PhD degree in computer science from the Institute of Computing Technology, Chinese Academy of Science, in 1995. He is a professor and the director of the Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, China. He visited the United Nations University International Institute for Software Technology as a research fellow in 1998 and 1999, respectively. From 1999 to 2000, he held a visiting position in the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. He was a visiting professor at the University of Texas, Arlington, from July to September 2004 and a visiting professor at the Hong Kong University of Science and Technology from December 2006 to February 2007. His current research focuses on multimedia systems and networking, sensor networks, and grid computing. He has published more than 100 papers and three books on these fields. He is a member of the IEEE and the ACM.